Design Verification

Lecture 17 - Model Checking II

1. Ad-hoc model checking approach **Example 1**

PS	**NS**	
	x=0	**x=1**
A	B	D
B	A	B
C	D	A
D	D	C

2. Model checking using CTL

   Some sample CTL formulas to be checked:

   - $EX((a = 1) \wedge (b = 1))$
   - $E((a = 1)U(b = 1))$
   - $EG((a = 1) \wedge (b = 1))$

   $\rightarrow$ The formal basis for proving a formula holds in a temporal structure is **fix-point characterization** of temporal operators. Alternatively, given a formula ϕ, determine **all** states that satisfy ϕ.

   $\rightarrow$ Every operator of the propositional temporal logic can be formally defined by a fixpoint equation. The idea is to split the formalization into two parts. In the first part, we only make propositions with regard to the actual state, and the second part makes propositions on the immediate successor states, using AX and EX.

For example, $AG\phi = \phi \wedge AX(AG\phi)$. In other words, $AG\phi$ says that ϕ must hold in the current state, and all successor states this formula $AG\phi$ must hold again and again.

Model-Checking fixpoint functions:

- atomic_functions $check\_af(a)$: if the atomic formula a is $q_i = 0$, then return all states whose output $q_i = 0$.
- check_EX(A): A is a set of states.
  $Z = \{s \mid \exists a \textit{ transition under which } s \textit{ goes to some state in } A\}$
  Returns all *predecessor* states in A. In other words, these are the states for which there is a single transition that takes the FSM to a state in A.
- check_EU(A, B): (checking for E $(\phi_1 U \phi_2)$). Both A and B are sets of states; A is the set that satisfies ϕ_1, and B is the set that satisfies ϕ_2. The resulting set is built iteratively:
  $Z_0 = B$
  $Z_1 = Z_0 \bigcup (A \bigcap check\_EX(Z_0))$
  $Z_2 = Z_1 \bigcup (A \bigcap check\_EX(Z_1))$
  ...
  $Z_k = Z_{k-1} \bigcup (A \bigcap check\_EX(Z_{k-1}))$
  Will this ever converge?
  $\rightarrow$ Yes, since for all k, although $Z_k \subseteq Z_{k+1}$, but because there are finitely many states, we must have some point where $Z_{k+1} = Z_k$. Worst case is Z_k containing all of S.
- check_EG(A): (checking for EG ϕ) A is a set of states. Again we compute this iteratively:
  $Z_0 = A$
  $Z_1 = Z_0 \bigcap check\_EX(Z_0)$
  $Z_2 = Z_1 \bigcap check\_EX(Z_1)$
  ...
  $Z_k = Z_{k-1} \bigcap check\_EX(Z_{k-1})$
  Will this converge? $\rightarrow$ Use similar analogy as before. Essentially, check_EG is based on decomposing the graph into strongly connected components (SCCs), where SCCs are non-trivial (eg. more than 1 node).
- How about computing check_AX, check_AG, etc?
  $\rightarrow$ recall that $A\phi \leftrightarrow \neg E(\neg\phi)$

Example 2

Example 3

Example 4

3. Complexity of check functions: (assuming there are n states and m edges
   (a) atomic formula: $O(m)$
   (b) $\neg\phi$: $O(n)$
   (c) $(\phi_1 \wedge \phi_2)$: $O(n)$
   (d) $EX\phi$: $O(m)$
   (e) $E\phi_1 U \phi_2$: $O(m)$
   (f) $EG\phi$: $O(n \times m)$

   The complexities look fine, but if the STG size is exponentially large (2^{40} states), and if it takes 2^{-10} seconds per states, then it would still require 2^{30} seconds or millions of hours!

   Solution: Use efficient methods to represent a Boolean function, such as OBDD's.

4. Remark: counter-example given if property not satisfied. (state where property violated is found)

5. Fair CTL

   $\rightarrow$ A fair temporal structure is a temporal structure (S, R, L) containing some sets $B_i \subseteq S$: $M = (S, R, L, B_1, ..., B_n)$

   Given a fair structure, then a path ψ is called **fair** with regard to the fairness constraints B_i, if it holds that $\forall B_i, B_i \cap \{s | s\ is\ visited\ infinitely\ often\} \neq \emptyset$.

   Alternatively, a fair path visits at least one state of each set B_i infinitely often. And usually, the sets B_i are not given explicitly by state sets by implicitly by fairness constraints ξ_i, specified in CTL. Thus, $B_i = \{s | s \models \xi_i\}$.

6. Model Checking with Fairness constraints

   A formula $EX\phi$ with fairness constraints ξ_i is true in a state s, if and only if there exists an immediate successor state s' such that $s' \models \phi$ and s' is the starting state of a fair path involving ξ_i.

   Example 5

   Thus, check_Fair_EX(A) = check_EX($A \wedge S_{fair}$), where each state in S_{fair} is a starting state of a fair path.

   But how do we compute S_{fair} given a CTL formula ϕ and a set of fairness constraints $\{\xi_0, \xi_1, ..., \xi_n\}$?

   $\rightarrow$ find states which lie at the beginning of infinite paths satisfying the fairness constraints that also are found wholly in A:

   - Let A = set of states that satisfies ϕ. Again, we can build S_{fair} iteratively, for all i's (for all constraints):

     $Z_0 = A$
     $Z_1 = Z_0 \cap check\_EX(check\_EU(A, Z_0 \cap \xi_i))$
     $Z_2 = Z_1 \cap check\_EX(check\_EU(A, Z_1 \cap \xi_i))$
     ...
     $Z_k = Z_{k-1} \cap check\_EX(check\_EU(A, Z_{k-1} \cap \xi_i))$

Interestingly, computing S_{fair} is the same as computing check_Fair_EG(A, B), where B is the set of fairness constraints. i.e., $S_{fair} = \mathit{check\_Fair\_EG}(A, B)$, where $A =$ set of states satisfying formula ϕ, and $B =$ set of fairness constraints ξ_i's.

Finally, computing $E(\phi U \psi)$ with fairness can be done analogously:

check_Fair_EU(A, B): A is the set that satisfies ϕ, and B is the set that satisfies ψ.

$\rightarrow$ check_Fair_EU$(A, B) =$ check_EU$(A, B \cap S_{fair})$

7. LTL checking method

   - Since $A\ \psi \equiv \neg E \neg \psi$, we just need to check for $E\ f$, where f is a path formula along linear temporal tree
   - basic idea: convert an LTL formula to a state diagram and check for infinite paths containing the property

8. state machine conversion: tableaux method

   - splitting rules for propositional formulas:
     $\longmapsto \phi = \phi_1 \wedge \phi_2$, tableau node $= \{\phi_1, \phi_2\}$ $\longmapsto \phi = \phi_1 \vee \phi_2$, tableau node $= \{\phi_1\}$, $\{\phi_2\}$
   - splitting rules for temporal formulas:
     Recall from fixed-points, we have:
     ϕ_1 U $\phi_2 \equiv fp_{min}(Z,\ \phi_2 \vee (\phi_1 \wedge XZ))$
     $F\ \phi \equiv\ true\ U\ \phi$
     $G\ \phi \equiv fp_{max}(Z,\ \phi \wedge X(G\ \phi))$
     Thus, for $\phi_1\ U\ \phi_2$, tableau $= \{\phi_2\}$, $\{\phi_1 \wedge X(\phi_1\ U\ \phi_2)\}$
   - for $X\ \phi$, a successor tableau node is formed with only ϕ
   - Note: a new node is added if it has not been created before

Example 6

Example 7

Example 8

Example 9

Example 10